Föreläsning

Reduktioner

Kan användas till att

- visa positiva resultat (hitta algoritmer)
- visa negativa resultat (undre gränser)
- visa att ett problem är svårare än ett annat
- visa att två problem är lika svåra

Olika varianter av problem

Olika varianter av reduktioner

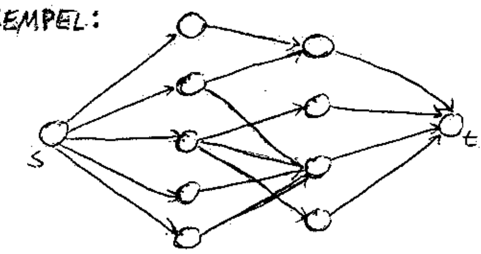
Bipartit matchning

Indata: bipartit graf $\langle U \cup V, E\rangle$

Utdata: en matchning $M \subseteq E$ av maximal storlek

M är en matchning om inga kanter i M har någon gemensam ändpunkt.

Exempel:



Reduktion av bipartit matchning till flöde:

Bipartite-Matching$(U,V,E)=$

- konstruera graf $\langle V', E'\rangle$ som

  $V' = U \cup V \cup \{s,t\}$

  $E' = \{(u,v):\ u \in U, v \in V, (u,v) \in E\} \cup$

  $\cup \{(s,u):\ u \in U\} \cup$

  $\cup \{(v,t):\ v \in V\}$

  $c(e) = 1\ \forall e \in E'$
- return (Ford-Fulkerson$(V',E') \cap E$)

Representantproblemet

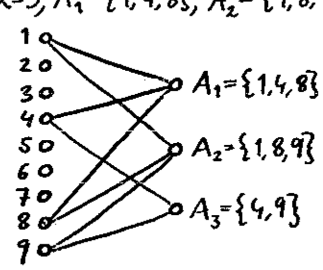
Indata: k stycken mängder $A_1, \dots, A_k$ av heltal mellan 1 och k^2

Utdata: en mängd med k olika tal $a_1, \dots, a_k$ så att $a_i \in A_i$ för $1 \le i \le k$

Algoritm: **reducera problemet till bipartit matchning!**

```
Representant(A_1,...,A_k) =
   U ← {1,...,k²}
   V ← {A_1,...,A_k}
   E ← {(x,A): x∈U, A∈V, x∈A}
   M ← BipartitMatchning(U,V,E)
   return {x: (x,A)∈M för något A}
```

Exempel: $k=3$, $A_1=\{1,4,8\}$, $A_2=\{1,8,9\}$, $A_3=\{4,9\}$



Generell viktad matchning

Det finns en polynomisk algoritm som löser Maximal maxmatchning:

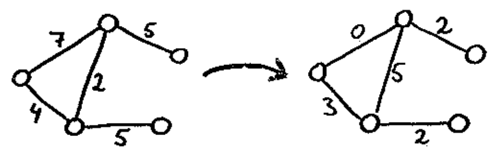
Indata: graf $G=(V,E)$, kantvikter $c: E \to \mathbb{N}$

Utdata: matchning $M \in E$ som dels innehåller så många kanter som möjligt och dels har så stor sammanlagd vikt som möjligt.

Beskriv en algoritm för problemet Minimal maxmatchning där sammanlagda vikten av kanterna i maxmatchningen är minimal!

Reducera Minimal maxmatchning till Maximal maxmatchning

```
MinimalMaxmatchning(G'=(V',E'), f: E'→ℕ) =
   w ← max f(e)
       e∈E'
   definera c(e) som w−f(e)
   M ← MaximalMaxmatchning(G', c)
   return M
```



Max klick

Indata: graf G

Utdata: antal hörn i den största klicken (fullständiga delgrafen) i G

Max oberoende mängd

Indata: graf G

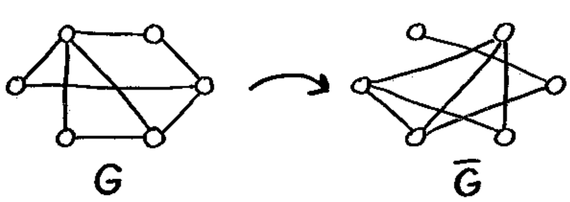
Utdata: antal hörn i den största oberoende mängden (hörn utan kanter mellan) i G

Reducera Max klick till Max oberoende mängd!

```
MaxKlick(G) =
  Ḡ ← komplementgrafen till G
  return MaxOberMängd(Ḡ)
```

Reducera Max oberoende mängd till Max klick!

```
MaxOberoendeMängd(G) =
  Ḡ ← komplementgrafen till G
  return MaxKlick(Ḡ)
```



Max klick och Max oberoende mängd är lika svåra!

Visa negativa resultat med reduktion

```
Sortering(A[1..n]) =
   T ← ByggSökträd(A)
   index ← 1
   Inorder(T)
   return A

Inorder(T) =
   if T≠NIL then
      Inorder(T.left)
      A[index] ← T.value
      index ← index+1
      Inorder(T.right)
```

Vi har reducerat sortering till sökträdsbygge.

Tiden för Sortering är tiden för ByggSökträd plus tiden för Inorder (som är $O(n)$).

Om ByggSökträd gick snabbare än $\Theta(n \log n)$ skulle det alltså gå att sortera snabbare än $\Theta(n \log n)$.

Reduktionen har fört över den undre gränsen på $\Omega(n \log n)$ jämförelser för sortering till sökträdsbygge.

> Om P kan reduceras till Q
> och P är svårt
> så är Q också svårt!

Olika varianter av problem

1. **Beslutsproblem** (utdata är sant/falskt)
   Indata: Graf G, heltal K
   Fråga: Finns det en klick av storlek $\geq K$ i G?

2. **Optimeringsproblem** (maximera en målfunktion)
   Indata: Graf G
   Utdata: Storleken av största klicken i G

3. **Konstruktionsproblem**
   Indata: Graf G
   Utdata: Hörnen i en klick av maxstorlek i G

Reduktioner mellan problemvarianter

Reducera optimeringsproblem till beslutsproblem:

```
MaxKlick(G=(V,E)) =
  i←1; j←|V|
  WHILE i<j DO
    m←⌈(i+j)/2⌉
    IF BeslutKlick(G,m) THEN i←m
      ELSE j←m-1
  RETURN i
```

Reducera konstruktionsproblem till optimeringsproblem:

```
KonstruktivMaxKlick(G=(V,E)) =
  m←MaxKlick(G)
  S←V
  FOR EACH u∈V DO
    IF MaxKlick(S-{u})=m THEN
       S←S-{u}
  RETURN S
```

Turing- och karpreduktioner

En turingreduktion av problemet P till problemet Q är en algoritm för P som anropar en algoritm för Q en eller flera gånger.

```
P(x)=
  ⋮
  (här anropas algoritm för Q)
  ⋮
  RETURN y
```

En karpreduktion av P till Q är det specialfall av turingreduktion där P och Q är beslutsproblem, där algoritmen för Q bara anropas en gång, och där resultatet från den algoritmen direkt returneras

```
P(x)=
  KONSTRUERA EN PROBLEM-
  INSTANS y TILL Q
  RETURN Q(y)
```